



# May 2000 Prototype Framework Tutorial: Examples

Paolo Calafiura

HCG/NERSC/LBNL

*ATLAS Software WorkShop @ LBNL*

*May 9, 2000*





# Example 1: HelloWorld Concepts



- **Part 0:**
    - set up build environments
    - create a new package
    - compile and link
    - run
  - **Part 1:**
    - The Messaging Service
  - **Part 2:**
    - The JobOptions and Properties Service
- 
- Files:
    - **HelloWorldcxx, HelloWorld.h**





# Example 1: Part 0

## Setup Build Environments



- > **setup atlas**
- > **setup gaudi??**
- > **mkdir Work; cd Work**
- > **mkdir src41; cd src41**
- > **source /auto/atlas/tools/tutorial\_setup**
- > **mkdir build41; cd build41**
- > **../src41/configure**
- > **gmake install**
- > **cd ..; mkdir run; cd run**
- **link data files, copy jobOptions, atlas.datback**
- **> tutorial\_examples < atlas.datback**





# Example 1: Part 1

## The Messaging Service



- **Need to include Message Service header:**

```
#include "Gaudi/MessageSvc/MsgStream.h"
```

- **To print to message service:**

```
MsgStream log(messageService(), name());  
log << MSG::INFO << "hello world" << endreq;
```

- **Various message levels:**

MSG::DEBUG

MSG::INFO

MSG::WARNING

MSG::ERROR

MSG::FATAL





# Example 1: Part 1

## Messaging Service (cont)



- add Message Service output level to jobOptions.txt

```
MsgService.OutputLevel = 2;
```
- Can set output level for each algorithm independently





# Example 1: Part 1

## Accessing Services from Algorithms



- Any registered service can be accessed from an algorithm once its header file has been included in the algorithm.
- The algorithm class provides hooks to the various services.
- The most common services are:

messageService()

eventDataService()

histogramDataService()

ntupleService()

detDataService()

serviceLocator()

and more. . . .





# Example 1: Part 2

## Properties



- Algorithms can have properties, which are settable at run time via the jobOptions file.
- Include the header for the Property Manager

```
#include "Gaudi/JobOptions/PropertyMgr.h"
```
- Declare the properties in initialize()

```
int m_foo; // in class header  
declareProperty("Foo", m_foo);
```
- This makes an association between a data member and a property name as used in the jobOptions
- The supported data types are:

```
int, double, bool, string  
vector<int>, vector<double>, vector<bool>  
vector<string>
```





# Example 1: Part 2

## Properties (cont)



- Add the property entry in the jobOptions file:  
`HelloWorld.Foo = 42;`
- Do the same thing for a bool, a double, and a string vector.
- Print out the value of the properties using the messaging service in the execute() method.





## Example 2: Transient Data Store, Multiple Algorithms



- Concepts:
- Part 1:
  - Getting to the Transient Data Store
  - Getting the event header from the TDS, and accessing its components
- Part 2:
  - Creating a DataObject
  - Writing DataObject to TDS
  - Reading DataObject from TDS
  - Chaining multiple algorithms





## Example 2: Transient Data Store, Multiple Algorithms



- Part 3:
  - Creating a ContainedObject
  - Writing a vector of ContainedObjects to TDS
  - Reading vector of ContainedObjects from TDS

### Files:

- **ReadData.cxx, ReadData.h**
- **WriteData.cxx, WriteData.h**





# Example 2: Part 1

## Getting to the TDS



- Start with the **ReadData algorithm skeleton**
- Add the headers for the **ZebraTDREvent, the event data service, and the Smart Data Pointer:**

```
#i ncl ude "ZebraTDRcnv/ZebraTDREvent.h"  
#i ncl ude "Gaudi /DataSvc/EvtDataSvc.h"  
#i ncl ude "Gaudi /DataSvc/SmartDataPtr.h"
```

- In the **execute()** method of **ReadData**, Get a **SmartDataPtr** to the Event in the TDS:

```
SmartDataPtr<ZebraTDREvent>  
evt(eventDataService(), "/Event");
```

- Check to see if a valid event was found:

```
if (!evt) { print error message}
```

- Get the event and run numbers:

```
event = evt->event_number();  
run = evt->run_number()
```





## Example 2: Part 2

### Creating a DataObject



- To be registered in the TDS, an object must be a **DataObject**, ie it must inherit from **DataObject**.
- You must also define a static constant Class ID, which must be unique for each class.

```
#include "Gaudi/Kernel/DataObject.h"
const static CLID CLID_MYDATAOBJECT = 214222;
class MyDataObject: public DataObject {
public:
    MyDataObject(): DataObject(){}
    const static CLID& classID() { return
CLID_MYDATAOBJECT; }
};
```





## Example 2: Part 2

### Writing DataObject to TDS



- The DataObject will be written to a location in the TDS as specified by a property in jobOptions:

```
decl areProperty("DataObj Location",
    m_DataObj Location);
```

- Add this to both the ReadData and WriteData algorithms.
- In the execute() method of WriteData, create a DataObject of type MyDataObject, and set its member data:

```
MyDataObj *dobj = new MyDataObj;
```

- Register it in the event TDS

```
StatusCode sc = eventDataService() ->
    registerObject(m_DataObj Loc, dobj);
if ( sc.isFailure() { print error message }
```





# Example 2: Part 2

## Read DataObject from TDS



- Objects in the TDS can be accessed via a **SmartDataPtr<>**, which is templated in the type of the object, and has a usage syntax similar to a normal pointer.
- In the **execute()** method of **ReadData**, get a **SmartDataPtr** of type **MyDataObj** from the event **TDS**, with its location specified by the **variable**:

```
SmartDataPtr<MyDataObj>
    dobj (eventDataService(), m_DataObjLoc);
    if (!dobj) { print error message }
```

- Print value of object member data:

```
log << MSG::INFO << "Data Object Val: " << dobj->Val() << endreq;
```





## Example 2: Part 2

### Chaining Multiple Algorithms



- In order to execute more than one algorithm, all that must be modified is the “TopAlg” property of the Application Manager in the jobOptions file:

```
ApplicationMgr. TopAlg = { "WriteData",  
    "ReadData" } ;
```

- Modify the **GNUmakefile.in** so that **WriteData.cxx** gets compiled
- Compile and link using “gmake install”
- run the executable in the run directory





## Example 2: Part 3

### Creating a Contained Object



- If you want to store a number of objects of the same type in the TDS, it is useful to store them in a special container, such as an `ObjectVector`.
- Objects must inherit from `ContainedObject` if they are to be stored in an `ObjectVector`:

```
#include "Gaudi/Kernel/ContainedObject.h"
#include <vector>
const static CLID CLID_MYCO = 214223;
class MyContObj : public ContainedObject {
public:
    MyContObj(): ContainedObject(){}
    const static CLID& classID() { return CLID_MYCO; }
    void IVec(vector<int> &v) { m_iv = v; }
    vector<int> IVec() const { return m_vec; }
private:
    vector<int> m_iv;
};
```





## Example 2: Part 3

### Writing a Contained Object



- In the `execute()` method of `WriteData`, create and fill a `ContainedObject` with the event and run number:

```
vector<int> iv;  
iv.push_back(run);  
iv.push_back(event);  
MyContObj *mco = new MyContObj;  
mco->ivec(iv);
```

- Create an `ObjectVector` on the heap, and fill it with a contained object:

```
ObjectVector<MyContObj> *cobj = new  
ObjectVector<MyContObj>;  
cobj->push_back(mco);
```





## Example 2: Part 3

### Writing a Contained Object



- Register the object **vector** with the TDS, and check to see that all went well:

```
sc = eventDataService() ->  
    registerObject(m_ContObjLoc, cobj);  
if ( sc.isFailure() ) { print error mesg };
```





# Example 2: Part 3

## Reading Contained Object from TDS



- In the `execute()` method of `ReadData`, get a `SmartDataPtr` that points to the contained object in the TDS:

```
SmartDataPtr< Obj ectVector<MyContObj > >
    vec(eventDataServi ce(), m_ContObj Loc);
```

- Iterate through the vector, printing out the values of its members:

```
Obj ectVector<MyContObj >:: i terator it;
for (it=vec->begin(); it!=vec->end(); it++) {
    vector<int> i vec = (*it)->I Vec();
    vector<int>:: const_i terator it_i ;
    for (it_i = i vec. begin(); it_i != i vec. end(); it_i++) {
        log << MSG::INFO << " " << (*it_i) << endreq;
    }
}
```

- Now compile and run the executable





# Example 3

## Sub-Algorithms



- Concepts:
  - Part 1:
    - Using sub algorithms
  - Part 2:
    - Using multiple identical sub-algorithms
- 
- Files
  - MainAlg.hxx, MainAlg.h
  - SubAlg.hxx, SubAlg.h





# Example 3: Part 1

## Sub-Algorithms



- **Sub-algorithms are not automatically created by the framework - this has to be done explicitly by their parent algorithm.**
- **Once created, the initialize() method of the sub-algorithm is automatically called, but the execute() phase must be explicitly invoked by the parent algorithm.**
- **The finalize() method is automatically called by the framework.**





# Example 3: Part 1

## Sub-Algorithms



- **The sub-algorithm will be referenced via a string property of the parent algorithm (MainAlg).**

```
decl areProperty("SubAI gType", m_subAI gType);  
decl areProperty("SubAI gName", m_subAI gName);
```

- **The properties of the sub-algorithm will also be set by properties of the parent algorithm**

```
decl areProperty("SubAI gPropName",  
    m_subAI gPropName);  
decl areProperty("SubAI gPropVal ", m_subAI gPropVal );
```

- **In order to identify the sub-algorithm, we want to keep a pointer to it as a member data of the parent algorithm. In the header of MainAlg:**

```
AI gori thm* p_Sub;
```





# Example 3: Part 1

## Sub-Algorithms



- In the **initialize()** method of the parent algorithm, the sub-algorithm must be created:

```
StatusCode sc;  
sc = createSubAlg(m_subAlgType,  
m_subAlgName, p_Sub);  
if ( sc.isFailure() ) { print error message }
```

- In the **execute()** method of the parent algorithm, we want to set a property of the sub-algorithm:

```
sc = p_Sub->setProperty( StringProperty  
(m_subAlgPropName, m_subAlgPropVal) );
```

- Then we want to execute the sub-algorithm:

```
sc = p_Sub->execute();
```





## Example 3: Part 2

### Multiple Identical Sub-Algorithms



- In order to execute multiple identical sub-algorithms, each one must be created by the parent algorithm with a different name:

`m_subAlg1 gori thm`

